(12) UK Patent Application (19) GB (11) 2 328 047 (13) A

(43) Date of A Publication 10.02.1999

(21) Application No 9813901.7

(22) Date of Filing 26.06.1998

(30) Priority Data
(31) 08884864 (32) 30.06.1997 (33) US

(51) INT CL[6]
G06F 1/00 12/14

(52) UK CL (Edition Q)
G4A AAP

(56) Documents Cited
EP 0820017 A2 EP 0717339 A2 EP 0456386 A2
EP 0442839 A2

(58) Field of Search
UK CL (Edition P) G4A AAP AMX
INT CL[6] G06F 1/00 12/14 , H04L 9/00 9/30
Selected publications and Online: COMPUTER, WPI

(71) Applicant(s)
Microsoft Corporation
(Incorporated in USA - Washington)
One Microsoft Way, Redmond, WA 98052-6399,
United States of America

(72) Inventor(s)
Matthew W Thomlinson
Scott Field
Allan Cooper

(74) Agent and/or Address for Service
Withers & Rogers
4 Dyer's Buildings, Holborn, LONDON, EC1N 2JT,
United Kingdom

(54) Abstract Title
Protected storage of core data secrets

(57) The invention provides central storage for core data secrets, referred to as data items. The architecture includes a storage server (104), a plurality of installable storage providers (106), and one or more authentication providers (108). Programming interfaces are exposed so that application programs (102) can utilize the services provided by the invention without having to actually implement the features. When storing a data item using the protected storage services, an application program can specify rules that determine when to allow access to the data item. Access can, if desired, be limited to the current computer user. Access can similarly be limited to specified application programs or to certain classes of application programs. The storage server authenticates requesting application programs before returning data to them. A default authentication provider authenticates users based on their computer or network logon. A default storage provider allows storage of data items on magnetic media such as a hard disk or a floppy disk. Data items are encrypted before they are stored. The encryption optionally uses a key that is derived from the previous authentication of the user. Specifically, the key is derived from the user's password, supplied during logon. In addition, an application program or the user can specify that certain items require another password that is entered whenever access to the data is requested. The default storage provider implements a multi-level encryption scheme to minimize the amount of encryption that has to be re-done when the user changes a password. Each data item is encrypted using an item key that is generated randomly by the system. The item key is in turn encrypted with a master key that is itself encrypted with a key derived from the user-supplied password (such as the user's logon password).

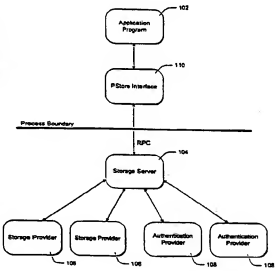

Fig. 2

GB 2 328 047 A

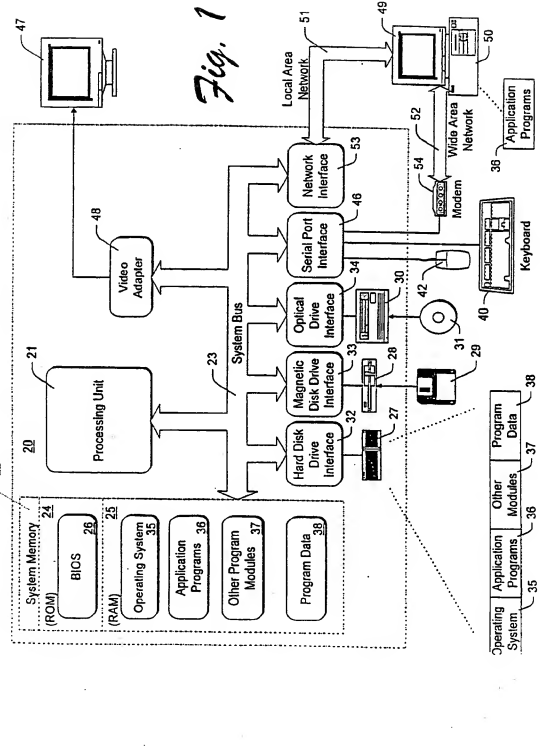
22
System Memory
20
(ROM)
24
BIOS
26
(RAM)
25
Operating System
35
Application Programs
36
Other Program Modules
37
Program Data
38
Processing Unit
21
23
System Bus
Hard Disk Drive Interface
32
Magnetic Disk Drive Interface
33
Optical Drive Interface
34
Video Adapter
48
Network Interface
53
Serial Port Interface
46
Local Area Network
51
Wide Area Network
52
Modem
54
Keyboard
40
Application Programs
38
Program Data
38
Other Modules
37
Application Programs
36
Operating System
35
47
49
50
27
28
29
30
31
42

Fig. 1

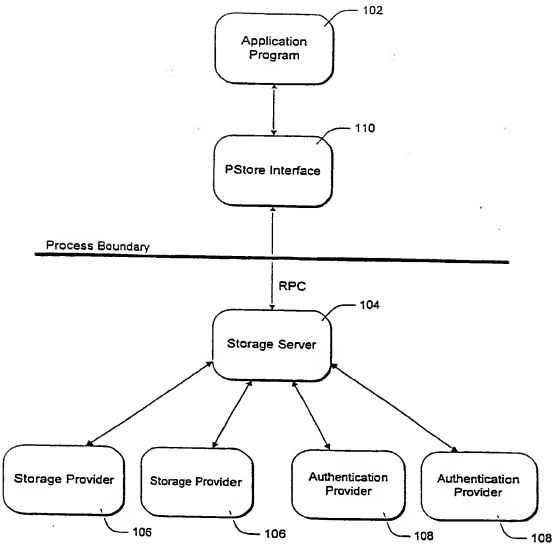
102
Application Program
110
PStore Interface
Process Boundary
RPC
104
Storage Server
Storage Provider
Storage Provider
Authentication Provider
Authentication Provider
106
106
108
108

Fig. 2

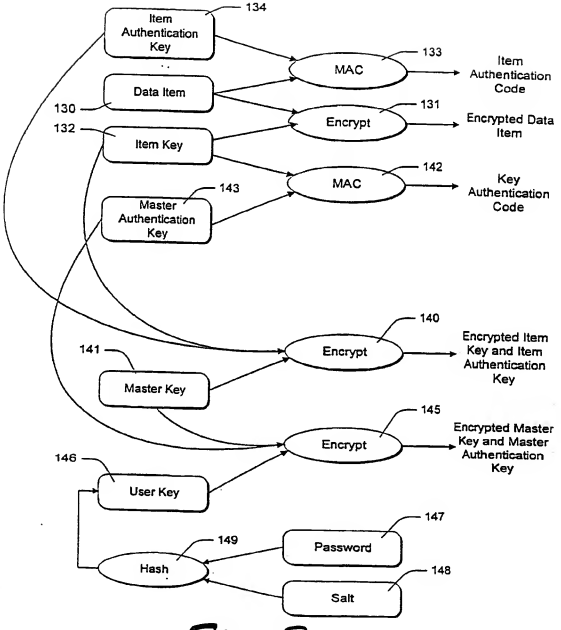
Item Authentication Key
134
MAC
133
Item Authentication Code
Data Item
130
Encrypt
131
Encrypted Data Item
132
Item Key
MAC
142
Key Authentication Code
143
Master Authentication Key
140
Encrypt
Encrypted Item Key and Item Authentication Key
141
Master Key
145
Encrypt
Encrypted Master Key and Master Authentication Key
146
User Key
147
Password
149
Hash
148
Salt

Fig. 3

2328047

PROTECTED STORAGE OF CORE DATA SECRETS

TECHNICAL FIELD

This invention relates to systems and methods that provide central services for securely storing core data secrets such as passwords, cryptographic keys, and sensitive personal or financial codes.

BACKGROUND OF THE INVENTION

Increasingly, financial and other sensitive transactions are being performed by personal computers. This has increased the need for secure storage of data. Modern cryptography techniques are often used to protect such data from unauthorized access.

New security methods, however, have brought about a need to store and protect "core" data secrets, such as private cryptographic keys, credit card numbers, and other small pieces of secret data. Presently, this responsibility is left to individual application programs or to personal computer users themselves. Although programs are available that allow users to encrypt and store data, such programs cannot typically be used by other application programs. Currently, each application program has to devise a safe and secure method to store such data.

As an example of the problems associated with the current state of the art, consider the issues involved in exploiting smart card technologies. A smart card is particularly well suited as a receptacle for core data secrets such as those described above. In addition, smart cards can be used to authenticate users by requiring each user to insert his or her personal smart card into a receptacle associated with the user's personal computer. Tamper-proof smart cards have been designed for just these purposes.

Problems arise without agreed upon standards for using such devices. Although a developer could provide capabilities for working with a limited number of smart cards, it would be difficult or impossible to anticipate all the different variations that might eventually arise. This fact makes it impractical to implement smart card technology in various different applications.

Although some storage media such as magnetic hard disks do not present the challenges of smart cards, many software developers simply do not have the background and knowledge required to safely implement modern cryptographic techniques. Even if they did, it would be inefficient for each developer to undertake the complex task of developing a method of storing core secrets. Furthermore, resulting solutions would be incompatible. It would be much more preferable to adopt a common scheme for storing such data, and to avoid having to implement a new solution for every different application program.

SUMMARY OF THE INVENTION

The invention described below provides central protected storage services that can be called by application programs to store core secrets. An embodiment of the invention is implemented as a server process and associated interfaces that can be invoked by application programs to store and retrieve small data items.

The general method and architecture includes a storage server and a plurality of installable storage providers and authentication providers. Each storage provider is adapted to securely store data using a specific type of media, such as magnetic media or smart cards. Details of the storage medium are hidden from the calling application programs. Authentication providers are used to authenticate users by different methods, such as by requesting passwords, by

reading smart cards, by retinal scans, or by other ways that might be devised in the future. Again, authentication details are generally hidden from the calling application programs.

Application programs interact with the storage server through well-defined interfaces. A data item can be stored with a simple call to the storage server, and can be retrieved later with a similar call. All encryption, decryption, item integrity checks, and user authentication are performed by the storage server and its associated providers. Because of this, application programs can take advantage of advanced security features without adding complexity to the application programs themselves.

When storing a data item using the protected storage services, an application program can specify rules that determine when to allow access to the data item. Access is generally limited to the computer user that created the data item. Access can similarly be limited to specified application programs or to certain classes of application programs. The storage server authenticates requesting application programs before returning data to them.

A default authentication provider authenticates users based on their computer or network logon. Other authentication providers can also be installed.

A default storage provider allows storage of data items on magnetic media such as a hard disk or a floppy disk. Data items are encrypted before they are stored. The encryption uses a key that is derived from the authentication of the user. Specifically, the key is derived from the user's password, supplied during computer or network logon. In addition, an application program or the user can specify that certain items require an additional password to be entered whenever access to the data is requested.

The default storage provider implements a multi-level key encryption scheme to minimize the amount of encryption that has to be re-done when the user changes a password. Each data item is encrypted using an item key that is generated randomly by the system. The item key is in turn encrypted with a master key that is itself encrypted (as described below) with a key derived from the user-supplied password (such as the user's logon password).

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a block diagram of an exemplary computer system suitable for use in implementing the invention.

Fig. 2 is a block diagram of a protected storage system and a calling application program in accordance with the invention.

Fig. 3 is a process diagram illustrating how data items are encrypted and authenticated in accordance with the invention.

DETAILED DESCRIPTION

Cryptography Overview

In general, cryptography is the process for encrypting or scrambling messages such that the messages can be stored and transmitted securely. Cryptography can be used to achieve secure communications, even when the transmission media (for example, the Internet) is untrustworthy. Computer users also use cryptography to encrypt sensitive files, so that an intruder cannot understand them. Cryptography can be used to ensure data integrity as well as to maintain secrecy. It is also possible to verify the origin of data items using cryptography, though the use of using digital signatures. When using

cryptographic methods, only the cryptographic keys must remain secret. The algorithms, the key sizes, and file formats can be made public without compromising security.

Using data encryption, a data item can be scrambled so that it appears like random gibberish and is very difficult to transform back to the original data without a secret key. This message can consist of ASCII text, a database file, or any other data.

Once a data item has been encrypted, it can be stored on non-secure media or transmitted over a non-secure network, and still remain secret. Later, the message can be decrypted into its original form.

When a data item is encrypted, an *encryption key* is used. This is comparable to a key that is used to lock a padlock. To decrypt the message, a *decryption key* must be used. The encryption and decryption keys are often, but not always, the same key.

There are two main classes of encryption algorithms: *symmetric algorithms* and *public-key algorithms* (also known as *asymmetric* algorithms). Systems that use symmetric algorithms are sometimes referred to as *conventional*.

Symmetric algorithms are the most common type of encryption algorithm. They are known as symmetric because the same key is used for both encryption and decryption. Unlike the keys used with public-key algorithms, symmetric keys are frequently changed.

Compared to public-key algorithms, symmetric algorithms are very fast and, thus, are preferred when encrypting large amounts of data. Some of the more common symmetric algorithms are RC2, RC4, and the Data Encryption Standard (DES).

Public-key (asymmetric) algorithms use two different keys: the *public key* and the *private key*. The private key is kept private to the owner of the key pair, and the public key can be distributed to anyone who requests it (often by means of a certificate). If one key is used to encrypt a message, then the other key is required to decrypt the message.

Public-key algorithms are very slow—on the order of 1,000 times slower than symmetric algorithms. Consequently, they are typically used only to encrypt session keys. They are also used to digitally sign messages.

One of the most common public-key algorithms is the RSA Public-Key Cipher.

Digital signatures can be used to distribute an unencrypted data item, while allowing the recipients to be able to verify that the message comes from its purported sender and that it has not been tampered with. Signing a message does not alter the message, it simply generates a digital signature string that can either be bundled with the message or transmitted separately.

Digital signatures are generated by using public-key signature algorithms: a private key is used to generate the signature, and the corresponding public key is used to validate the signature.

Authentication involves the process of verifying the identity of a person or entity. Certificates are a common way to achieve authentication. A certificate is a set of data that completely identifies an entity, and is issued by a *Certification Authority* (CA) only after that Authority has verified that the entity is who it says it is. The data set includes the entity's public cryptographic key. When the sender of a message *signs* data with its private key (and sends a copy of its certificate with the message), the recipient of the message can use the sender's public key

(retrieved from the certificate) to verify that the sender is who it says it is. Certificates can also be used to verify that data (including application programs) have been vouched for by a trusted source.

On a network, there is often a trusted application running on a secure computer that is known as the Certification Authority. This application knows the public key of each user. Certification Authorities dispense messages known as certificates, each of which contains the public key of one of its client users. Each certificate is signed with the private key of the Certification Authority.

The invention described below utilizes techniques such as the well-known digital encryption, signing, and authentication techniques described above. For further information regarding such techniques, refer to Schneier, Bruce; *Applied Cryptography Second Edition: Protocols, Algorithms, and Source Code in C*; John Wiley & Sons, 1996, which is hereby incorporated by reference. The following discussion assumes general familiarity with these topics.

Exemplary Operating Environment

Fig. 1 and the following discussion are intended to provide a brief, general description of a suitable computing environment in which the invention may be implemented. Although not required, the invention will be described in the general context of computer-executable instructions, such as program modules, being executed by a personal computer. Generally, program modules include routines, programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types. Moreover, those skilled in the art will appreciate that the invention may be practiced with other computer system configurations, including hand-held devices, multiprocessor systems, microprocessor-based or programmable consumer electronics, network

PCs, minicomputers, mainframe computers, and the like. The invention may also be practiced in distributed computer environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computer environment, program modules may be located in both local and remote memory storage devices.

With reference to Fig. 1, an exemplary system for implementing the invention includes a general purpose computing device in the form of a conventional personal computer 20, including a processing unit 21, a system memory 22, and a system bus 23 that couples various system components including the system memory to the processing unit 21. The system bus 23 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. The system memory includes read only memory (ROM) 24 and random access memory (RAM) 25. A basic input/output system 26 (BIOS), containing the basic routines that help to transfer information between elements within personal computer 20, such as during start-up, is stored in ROM 24. The personal computer 20 further includes a hard disk drive 27 for reading from and writing to a hard disk, not shown, a magnetic disk drive 28 for reading from or writing to a removable magnetic disk 29, and an optical disk drive 30 for reading from or writing to a removable optical disk 31 such as a CD ROM or other optical media. The hard disk drive 27, magnetic disk drive 28, and optical disk drive 30 are connected to the system bus 23 by a hard disk drive interface 32, a magnetic disk drive interface 33, and an optical drive interface 34, respectively. The drives and their associated computer-readable media provide nonvolatile storage of computer readable instructions, data structures, program modules and other data

for the personal computer 20. Although the exemplary environment described herein employs a hard disk, a removable magnetic disk 29 and a removable optical disk 31, it should be appreciated by those skilled in the art that other types of computer readable media which can store data that is accessible by a computer, such as magnetic cassettes, flash memory cards, digital video disks, Bernoulli cartridges, random access memories (RAMs) read only memories (ROM), and the like, may also be used in the exemplary operating environment.

A number of program modules may be stored on the hard disk, magnetic disk 29 optical disk 31, ROM 24, or RAM 25, including an operating system 35, one or more application programs 36, other program modules 37, and program data 38. A user may enter commands and information into the personal computer 20 through input devices such as keyboard 40 and pointing device 42. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 21 through a serial port interface 46 that is coupled to the system bus, but may be connected by other interfaces, such as a parallel port, game port, or a universal serial bus (USB). A monitor 47 or other type of display device is also connected to the system bus 23 via an interface, such as a video adapter 48. In addition to the monitor, personal computers typically include other peripheral output devices (not shown) such as speakers and printers.

The personal computer 20 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 49. The remote computer 49 may be another personal computer, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to the personal

computer 20, although only a memory storage device 50 has been illustrated in Fig. 1. The logical connections depicted in Fig. 1 include a local area network (LAN) 51 and a wide area network (WAN) 52. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets, and the Internet.

When used in a LAN networking environment, the personal computer 20 is connected to the local network 51 through a network interface or adapter 53. When used in a WAN networking environment, the personal computer 20 typically includes a modem 54 or other means for establishing communications over the wide area network 52, such as the Internet. The modem 54, which may be internal or external, is connected to the system bus 23 via the serial port interface 46. In a networked environment, program modules depicted relative to the personal computer 20, or portions thereof, may be stored in the remote memory storage device. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

The illustrated computer uses an operating system such as the Windows family of operating systems available from Microsoft Corporation. The functionality described below is implemented using standard programming techniques, including the use of OLE and COM interfaces such as described in Brockschmidt, Kraig; *Inside OLE 2*; Microsoft Press, 1994, which is hereby incorporated by reference.

More recent Windows operating systems utilize what is referred to as the Win32 API: a well-defined set of interfaces that allow application programs to utilize functionality provided by the Windows operating systems. The Win32 API

is documented in numerous texts, including Simon, Richard; *Windows 95 Win32 Programming API Bible*; Waite Group Press, 1996, which is hereby incorporated by reference. General familiarity with this type of programming is assumed throughout the following discussion.

Overall Functionality and Architecture

Fig. 2 shows architectural components of a protected storage system in accordance with the invention for storing data items and for protecting them from unauthorized access. The protected storage system allows application programs to securely store data items that must be kept private and free from tampering. Such data items might include cryptographic keys, passwords, financial information, trust profiles, etc. The storage system is designed to hold small items of core secret data in a central and common storage location; if a lot of data is to be protected, a bootstrap secret (such as an encryption key) may be stored in the storage system rather than the data itself. This enables data items to be moved when appropriate to small, secure hardware devices such as smart cards, and also avoids unnecessary overhead which would otherwise be required to secure large data items.

The protected storage system is executed by a computer such as described above with reference to Fig. 1. Application programs, such as application program 102 shown in Fig. 2, are also executed by the computer.

Storage system 100 includes a storage server component 104, also referred to as a dispatcher. Generally, the storage server, through the Pstore interface 110, receives data items from application programs, securely stores the data items, and returns such data items in response to requests from application programs. The storage server also performs authentication and verification procedures with

respect to the requesting application programs, as will be explained in more detail below.

Associated with the storage server are a plurality of installable storage providers 106 and a plurality of installable authentication providers 108. Specified storage providers are called by the storage server to securely store and retrieve data items. One or more authentication providers are called by the storage server to identify and/or authenticate current computer users.

A default storage provider is implemented within the storage system. The default storage provider encrypts data items and then stores them on a hard disk or floppy disk. The encryption is based on one or more keys that are derived from authentication of the current computer user. The default storage provider also verifies the integrity of data items when retrieving and decrypting them.

A default authentication provider is also implemented within the storage system. The default authentication provider is called by the storage server to identify current computer users based on previously completed operating system logon procedures. Specifically, the default authentication provider identifies the current computer user based on the logon identification and password provided by the user while logging on to the computer's operating system or to a network operating system.

Although default storage and authentication providers are implemented, additional providers can be installed in the future to take advantage of new storage and authentication technologies. For example, a smart card storage provider might be installed to allow storage of core data items on a smart card. A smart card authentication provider might similarly be provided to authenticate users by

requiring them to insert their smart cards into a smart card reader. In either case, the smart card could utilize public-key cryptographic techniques.

Well-defined COM interfaces are used between the storage server and the various providers, allowing new providers to be easily installed and registered with the storage server. Application programs are able to make the same calls to storage server 102 regardless of which providers are used. Because of this, application programs can be written to take advantage of future technologies by simply interacting with storage server 102, and without having to concern themselves with the details of such future technologies.

To increase security, the protected storage system is implemented in a different address space than the calling application programs. Communications across the process or address space boundary take place using remote procedure calls (RPCs). Such calls can be made when operating under Windows operating systems and other operating systems. The functionality and formatting of RPC calls is documented in the Microsoft Win32 Software Development Kit.

Although application programs can make RPC calls directly, this complexity is avoided by providing a dynamically linked library (DLL) that can be executed in the application programs' address spaces. This library, referred to as Pstore Interface 110 in Fig. 2, implements a plurality of interfaces and associated methods that can be called by the application programs to exploit the full functionality of the protected storage system. The interfaces include methods for creating and reading data items, as well as other useful functions as described in an appendix to this document. The interfaces and methods in the described embodiment are implemented using the COM (component object model) interfaces of the Windows operating system.

Protected-Data Access Control

The protected storage system described herein has powerful data access control capability. The storage system offers two levels of data protection: application-level protection and user-level protection. At the application level, the storage server returns requested data items only to authorized requesting application programs. Furthermore, the storage server authenticates requesting application programs before returning individual data items. Application program authentication refers to a process of determining whether an application program has been tampered with, or to an alternative process of verifying a program's trustworthiness through use of public/private key cryptographic digital signatures or other means. Microsoft Authenticode is an available mechanism for verifying, through the use of digital signatures, that an application program has been published by a specified person, group, or organization, and that it is therefore trustworthy. Authenticode functionality is publicly available from Microsoft Corporation as part of its Win32 Software Development Kit.

In the embodiment describe herein, data items are organized in a hierarchical fashion by specifying types and subtypes as follows:

Type—Subtype—Data Item

There are predefined types and subtypes, and application programs can create new types and subtypes. Different protection levels can be established for data items falling under certain types and subtypes. Specifically, an *access rule set* is specified for each subtype. An access rule set contains rules for different types or subtypes of data items. Access to a data item of a particular subtype is granted if any single rule of the corresponding rule set is satisfied. Each access rule comprises a mode and one or more access clauses; all of the access clauses must

be satisfied before the rule is considered satisfied. The mode indicates the type of access allowed if all the clauses in a rule are satisfied. Currently, there are two access modes defined: *read* and *write* access.

There are currently three types of access clauses: *Authenticode*, *Binary Check*, and *Security Descriptor*. Authenticode verifies the application program requesting access to the protected data is trusted and can optionally determine whether the originator, and thus the originator's application, can be trusted. Binary Check ensures that a program has not been tampered with since installation. The Security Descriptor clause provides access based on Windows NT access control lists (ACLs).

Authenticode is a well-documented product and service available from Microsoft Corporation. If the Authenticode clause was specified at the time of subtype creation, a check is made to see if the requesting application was signed or not, and if signed, by whom. The clause may specify a particular root, certificate issuer, publisher (signer), program name, or some combination of the foregoing. Access will not be granted unless the specified criteria are met. If no criteria specified, the verification amounts to allowing any Authenticode-verified application or module access to the data. Authenticode checking also verifies the binary image of the module under inspection.

The Binary Check is implemented by taking a hash of a binary image of an application program at initialization. When the application program asks for data at a later time, the storage system again takes a hash of the memory image and compares it to the original hash. The two hashes must match before the protected storage system will allow the application program to access requested data. Thus, if the application has changed since it was installed, and therefore is likely to have

been tampered with, the data will be protected and access to the data by the application will be denied.

The Security Descriptor clause is intended to allow access only to specified users or groups, and is enforced on Windows NT platforms. This clause gets the necessary information about the users and groups from the ACLs contained in the Windows NT security descriptor.

At the user level, the storage server allows access to individual data items depending on the current computer user; by default, only the user responsible for creating the data item is allowed to access it. However, a user can override this default behavior by specifying appropriate options when creating and saving data items, or the user can later modify access rights to the data.

Users and application programs can specify security styles, which specify a degree and/or type of confirmation or authentication is required to access a particular data item; for instance, whether a password is required. The current embodiment, with the default authentication provider, supports the following access styles:

- Silent access: no user interaction required. Authentication is based on a previously completed computer or network operating system authentication procedure. In most cases, this type of authentication relies on the user being able to enter the correct password during a previous logon procedure, and no further interaction is required when protected data is actually accessed.
- Logon password: a dialog box is presented requiring the user to enter the password previously used to logon to the computer or network.

- User-defined password: the user specifies a password when an item is initially stored, and must enter the password before the data can be accessed again. Such passwords can be different for different data items, or groups of data items can be stored under the same password.
- OK/cancel: when an application attempts to access the data, a dialog box appears. The user responds to the dialog box by clicking on an OK or deny button, thereby granting/denying access to the data by a requesting application program.

As is apparent from the different types of access styles, accessing items in protected storage may require user interaction. This interaction is implemented through the use of a user alert dialog box. Typically, the user will be required to enter a password in response to a user alert dialog box. However, different authentication providers might require different types of responses (such as physical insertion of a hardware token or biometric authentication procedures).

To prevent attacking programs from presenting similar user alert dialogs, and thereby gaining access to secret data, the user alert dialogs can be customized by the user. Specifically, a user can specify a particular background or digital watermark to be used in the user alert dialog box. Alternatively, such a watermark can be randomly generated for the user. The user will become familiar with whatever watermark has been selected, and will thus recognize unauthorized dialog boxes generated by attacking applications.

Data Encryption, Decryption, and Authentication

Different storage providers may protect stored data in different ways. However, some type of cryptography will usually be employed. The default storage provider described herein uses a password-based encryption scheme,

wherein data items are encrypted based on a user-supplied password, or some other code related to user authentication, before storing the data items. When retrieving the data items, decryption is based on the same password or code.

When a data item is protected by the "user-defined password" security style mentioned above, the user explicitly enters a password for each data item during an authentication step that is invoked prior to allowing access to an individual data item. In the case of "silent access," however, encryption and decryption are based on a password or other code that is supplied by the current computer user during a previous computer or network operating system authentication or logon procedure. Typically, a user's logon name and password are used to form or derive a key that is used for encrypting and decrypting data items.

In the described embodiment, a multi-level key technique is used to encrypt data items based on user-supplied codes or passwords. This technique is illustrated in Fig. 3. In this implementation, encryption and decryption use one or more keys that are derived from the supplied passwords or logon codes. As mentioned, the password or code can either be gathered from a previous authentication step, or the storage system might prompt the current computer user for a password.

Generally, an item key is randomly generated for each data item. The data item is encrypted with its corresponding item key. An item authentication key is also generated randomly for each item and is used to generate an item authentication code. Item authentication codes are used during decryption to verify that data items are decrypted correctly.

The item key and item authentication key are then encrypted using a master key. The master key is a randomly generated number. A master authentication

key is also generated and used to calculate a key authentication code so that the correct decryption of the item key and item authentication key can be verified later. Finally, the master key and master authentication key are encrypted using a password that is derived from user authentication or identification.

With reference now to the specific steps of Fig. 3, an individual data item that is to be encrypted and stored is referenced by numeral 130. A step or operation 131 is performed of encrypting data item 130 using an item key 132. Specifically, cryptographic key 132 is used to perform a DES encryption on data item 130. Item key 132 is generated as a random number by the default storage provider.

The storage provider also performs a step 133 of generating an item authentication code for individual data item 130. The item authentication code is generated using a MAC (message authentication code) in conjunction with a randomly generated item authentication key 134. MACs are described in the Schneier text mentioned above.

A further step 140 is performed of encrypting the item key 132 and the item authentication key 134 with a master key 141, again using the DES encryption mentioned above. The master key is a random number. A step 142 comprises generating a key authentication code for the combination of the item key and the item authentication key. The key authentication code is generated with a MAC in conjunction with a randomly generated master authentication key 143.

A step 145 is performed of encrypting the master key and the master authentication key with a user key 146. This is again a DES encryption.

The user key is derived from the user-supplied password or code, referenced in Fig. 3 by numeral 147. To generate the user key, the user-supplied

password 147 is appended to a random number referred to as a salt 148, and hashed in a step 149 using an SHA-1 hashing function. This results in a number that is used as the user key.

Once these steps are performed, the storage server stores the encrypted individual data item, the item authentication code, the encrypted item key, the encrypted item authentication key, the key authentication code, the encrypted master key, and the encrypted master authentication key, to be retrieved later when requested by an authorized application program.

Retrieval comprises the reverse process. The encrypted items are retrieved from storage. The storage provider derives the user key from the user-supplied password and uses the user key to decrypt the master key and master authentication key. The master authentication key is used in conjunction with the specified MAC to verify that the master key decrypted correctly. The master key is then used to decrypt an appropriate item key and corresponding item authentication key. The item authentication key is used in conjunction with the MAC to verify that the item key decrypted correctly. The item key is then used to decrypt the actual data item.

This process allows all of a user's data items to be controlled by a single master key that is in turn encrypted as a function of the user's password. The advantage of this scheme is that data items do not have to be re-encrypted when the user changes his or her password. Rather, only the master key needs to be encrypted again.

Verification of Storage System Integrity

The storage server, the storage providers, and the authentication providers employ a security interlock mechanism to prevent security violations that might

result from tampering with system components. This mechanism utilizes cryptographic techniques.

One motivation for the security interlock mechanism is to prevent non-authorized providers from being loaded by the storage server. It is particularly important to prevent a non-authorized module from masquerading as an authorized provider, since such a non-authorized module could steal secret data from the system. Another motivation is to prevent tampering with the storage server itself.

When the server and providers are shipped, they are digitally signed with the private key of a public/private cryptographic key pair—the private key has a corresponding public key. The public key is then hard-coded into the various modules of the server and providers. The server and the providers are configured to verify each others' signatures using the public cryptographic key whenever an individual component is loaded and executed. When the server is loaded, it first checks its own integrity by checking its own digital signature with the public key. The server then checks the digital signatures of other core components as they are loaded. As each component is loaded, it checks the digital signature of the server. If any integrity check fails, none of the components will operate.

Authentication of Requesting Application Programs

As discussed above, access to data items can be restricted based on which application programs are trying to access the data items. For this feature to be reliable, the storage system needs to verify that application programs are who they say they are, and that they have not been tampered with. This process is referred to as program authentication. One option is to authenticate programs based on a binary check. Such an authentication is performed by two storage server modules: the *identification module* and the *enforcement module*.

The identification module is responsible for interrogating the client that is calling the storage server. In order to identify a process associated with a request, the following steps occur:

1. The client application program identifies itself to the server, presenting two pieces of information: a process ID, and a thread handle. The process ID is obtained using the GetCurrentProcessId() system call; the thread handle is obtained using the GetCurrentThread() and DuplicateHandle() system calls.
2. The storage server opens a handle to the client, using the process ID in a call to the system call OpenProcess(). The storage server saves this handle for later use.
3. The client makes access requests for data items.
4. The server uses the process handle obtained above to analyze the memory address space associated with the client process. The server also uses this handle to query the underlying operating system about what executable modules (.exe, .dll, etc. files) are present in the associated process, in addition to determining module load addresses; the exact method used to query the operating system varies depending on the operating system.
5. The server now has a complete list of modules associated with the client, and uses it to analyze the call stack associated with the thread handle obtained above. The StackWalk() system call is utilized to determine the chain of callers associated with the client.

The enforcement module uses results provided by the identification module in performing the following checks:

1. Verifying that the image loaded into the client process has not been tampered with on-disk. This is accomplished by storing a cryptographic representation of the file(s) that are to be granted access. This cryptographic representation is stored alongside the data. There can be two cryptographic representations of the file:
    - The entire file is read and then subjected to the SHA-1 cryptographic hash. The output of the hash is stored alongside the data. When subsequent access to the data is requested, the hash is recomputed against the on-disk file, and then compared to that stored alongside the data. If these compare correctly, the process continues to check 2, below.
    - The file is subject to public key certificate-based validation. This uses Microsoft Authenticode calls to verify that the image has not been tampered with. Authenticode handles hashing the disk image internally. This cryptographic representation of the file is more flexible, because it also supports validation against various fields in the certificate attached to the specified file. After the Authenticode verification takes place, the system analyzes the certificate contents, to make sure they match those that were stored alongside the data being accessed.
2. Verifying that the image on disk matches that loaded into the client process.
    - The module to be checked is "mapped" into the server address space, using the CreateFileMapping() and MapViewOfFile() system API calls.

- Relocation fixups are applied to the mapped image if necessary—only if the image did not load at the preferred address in the client address space.
- The system loops over the image header, looking for read-only sections such as code sections, resources, and read-only data. For each section, it updates an SHA-1-based cryptographic hash.
- The process handle output from the identification module is now used to read the memory address space where the module is loaded. This is accomplished by using the ReadProcessMemory() system call. Each section of memory is read in the manner outlined in the previous step, updating a cryptographic hash as the process proceeds.
- The system compares the two hashes resulting from the immediately preceding steps. If they match, the image in memory has not been tampered with.

Application Interface Functions

As described above, interfaces are exposed to application programs so that application programs can take advantage of protected storage features without having to implement sophisticated encryption schemes and without having to make RPC calls. These interfaces and their functions are described in an attached appendix that forms part of this document. The appendix also provides explanations regarding the proper usage of the interfaces.

Conclusion

The invention provides a versatile and efficient architecture that provides a number of advantages over the prior art. One significant advantage is that

different application programs can utilize a single, provided server to store core data secrets in a central storage area. This promotes consistency among the applications and removes significant overhead from the applications. The user interface is one area that benefits from the consistency provided by the storage system described above, since user prompts are generated by the system rather than by the individual application programs. Storing data items in a uniform manner also allows them to be managed by a single management program that is independent of the application programs themselves.

Another significant advantage of the invention is that the underlying details of securing data items are hidden from calling application programs. Thus, program developers do not have to implement sophisticated security measures; such measures can be implemented with simple calls to the storage system described herein. An added benefit is that new technologies such as smart cards will be available to application programs without extensive reprogramming.

The invention protects secrets from user-oriented and software-oriented attacks, including attacks from viruses. Significantly, access control is managed outside the application programs that generate and access data items. Because applications do not have direct access to keying material or other control data, access to one piece of data does not imply access to any other data. Furthermore, the storage system itself does not retain the information required to decrypt stored data items. Rather, the user must be present and must supply a correct password to allow data decryption.

A further important benefit of the invention is that users are not forced to explicitly enter passwords when data access is required. Rather, user authentication is performed once, when the user logs on to the computer or

network. This logon information is used for both user authentication and to derive keys for data encryption and decryption.

Although the invention has been described in language specific to structural features and/or methodological steps, it is to be understood that the invention defined in the appended claims is not necessarily limited to the specific features or steps described. Rather, the specific features and steps are disclosed as exemplary forms of implementing the claimed invention.

CLAIMS

1. A system for storing data items and for protecting them from unauthorized access, comprising:

a storage server that receives data items from application programs and that returns such data items in response to requests from application programs;

one or more storage providers that are called by the storage server and that securely store and retrieve the data items;

one or more authentication providers that are called by the storage server to identify current computer users.

2. A system as recited in claim 1, wherein the storage server is configured to call installable storage providers.

3. A system as recited in claim 1, wherein the storage server is configured to call installable authentication providers.

4. A system as recited in claim 1, wherein the storage server executes in a different address space than the application programs.

5. A system as recited in claim 1, wherein the storage server executes in a different address space than the application programs and is called via remote procedure calls.

6. A system as recited in claim 1, wherein said one or more authentication providers include an authentication provider that identifies users based on a previous network authentication.

7. A system as recited in claim 1, wherein said one or more storage providers include a storage provider that encrypts data items before storing them and decrypts data items before retrieving them.

8. A system as recited in claim 1, wherein said one or more storage providers include a storage provider that verifies the integrity of data items when retrieving them.

9. A system as recited in claim 1, wherein said one or more storage providers include a storage provider that encrypts and decrypts data items using one or more keys that are derived from authentication of the current computer user.

10. A system as recited in claim 1, wherein said one or more storage providers include a storage provider that encrypts and decrypts data items using one or more keys that are derived from a user-supplied code, and wherein the storage server prompts the current computer user for the user-supplied code.

11. A system as recited in claim 1, wherein said one or more storage providers include a storage provider that encrypts and decrypts data items using one or more keys that are derived from a network logon code supplied by the current computer user.

12. A system as recited in claim 1, wherein said one or more storage providers include a storage provider that securely stores and retrieves data items using one or more keys that are derived from a user-supplied code.

13. A system as recited in claim 1, wherein said one or more storage providers include a storage provider that securely stores and retrieves data items using one or more keys that are derived from a network logon code supplied by the current computer user.

14. A system as recited in claim 1, wherein said one or more storage providers include a storage provider that stores data items in accordance with the following steps:

encrypting individual data items with item keys;

encrypting the item keys with a master key;

encrypting the master key with a user key that is derived from a user-supplied code;

storing the encrypted individual data items, the encrypted item keys, and the encrypted master key.

15. A system as recited in claim 1, wherein said one or more storage providers include a storage provider that stores data items in accordance with the following steps:

encrypting individual data items with item keys;

generating item authentication codes for individual data items using item authentication keys;

encrypting the item keys and the item authentication keys with a master key;

generating key authentication codes for item keys and item authentication keys using a master authentication key;

encrypting the master key and the master authentication key with a user key that is derived from a user-supplied code;

storing the encrypted individual data items, the item authentication codes, the encrypted item keys, the encrypted item authentication keys, the key authentication codes, the encrypted master key, and the encrypted master authentication key.

16. A system as recited in claim 1, wherein said one or more storage providers include a storage provider that retrieves data items in accordance with the following steps:

retrieving encrypted individual data items, encrypted item keys corresponding to the encrypted individual data items, and an encrypted master key;

decrypting the encrypted master key with a user key that is derived from a user-supplied code:

decrypting the encrypted item keys with the decrypted master key;

decrypting the encrypted individual data items with the corresponding decrypted item keys.

17. A system as recited in claim 1, wherein the storage server allows access to individual data items depending on the current computer user.

18. A system as recited in claim 1, wherein the storage server returns requested data items only to authorized requesting application programs.

19. A system as recited in claim 1, wherein the storage server returns requested data items only to authorized requesting application programs, and wherein the storage server authenticates requesting application programs before returning individual data items.

20. A system as recited in claim 1, wherein the storage server returns requested data items only to authorized requesting application programs, and wherein the storage server authenticates the requesting application program using public key cryptography before returning individual data items.

21. A system as recited in claim 1, wherein the storage server is responsive to application program requests to present user dialogs allowing the current computer user to specify passwords to be used by the storage provider to securely store and retrieve data items.

22. A system as recited in claim 1, wherein the storage server, the storage providers, and the authentication providers are individually signed by a private cryptographic key that corresponds to a public cryptographic key; the storage server, the storage providers, and the authentication providers being configured to verify each others' signatures using the public cryptographic key.

23. A system for storing data items and for protecting them from unauthorized access, comprising:

a storage server that receives data items from application programs and that returns such data items in response to requests from application programs, wherein the storage server executes in a different address space than the application programs and is called via remote procedure calls;

a storage provider that is called by the storage server to securely store and retrieve the data items, wherein the storage provider encrypts data items before storing them using one or more keys that are derived from authentication of the current computer user, the storage provider verifying the integrity of data items when retrieving them;

an authentication provider that is called by the storage server to identify current computer users, wherein the authentication provider identifies users based on a previous operating system logon procedure.

24. A system as recited in claim 23, wherein said one or more keys are derived from an operating system logon code supplied by the current computer user.

25. A system as recited in claim 23, wherein said one or more keys are derived from a user-supplied code.

26. A system as recited in claim 23, wherein the storage provider stores data items in accordance with the following steps:

encrypting individual data items with item keys;

encrypting the item keys with a master key;

encrypting the master key with a user key that is derived from a user-supplied code;

storing the encrypted individual data items, the encrypted item keys, and the encrypted master key.

27. A system as recited in claim 23, wherein the storage provider stores data items in accordance with the following steps:

encrypting individual data items with item keys;

generating item authentication codes for individual data items using item authentication keys;

encrypting the item keys and the item authentication keys with a master key;

generating key authentication codes for item keys and item authentication keys using a master authentication key;

encrypting the master key and the master authentication key with a user key that is derived from a user-supplied code;

storing the encrypted individual data items, the item authentication codes, the encrypted item keys, the encrypted item authentication keys, the key authentication codes, the encrypted master key, and the encrypted master authentication key.

28. A system as recited in claim 23, wherein the storage provider retrieves data items in accordance with the following steps:

retrieving encrypted individual data items, encrypted item keys corresponding to the encrypted individual data items, and an encrypted master key;

decrypting the encrypted master key with a user key that is derived from a user-supplied code;

decrypting the encrypted item keys with the decrypted master key;

decrypting the encrypted individual data items with the corresponding decrypted item keys.

29. A system as recited in claim 23, wherein the storage server returns requested data items only to authorized requesting application programs.

30. A system as recited in claim 23, wherein the storage server returns requested data items only to authorized requesting application programs, and wherein the storage server authenticates requesting application programs before returning individual data items.

31. A system as recited in claim 23, wherein the storage server returns requested data items only to authorized requesting application programs, and wherein the storage server authenticates requesting application programs using public key cryptography before returning individual data items.

32. A system as recited in claim 23, wherein the storage server is responsive to application program requests to present user dialogs allowing the current computer user to specify passwords to be used by the storage provider to securely store and retrieve data items.

33. A system as recited in claim 23, wherein the storage server and the storage provider are individually signed by a private cryptographic key that corresponds to a public cryptographic key; the storage server and the storage provider being configured to verify each others' signatures using the public cryptographic key.

34. A method of storing user and application secrets and for protecting them from unauthorized access, comprising the following steps:

authenticating a current computer user based on a previous logon authentication performed by an operating system;

receiving individual data items from application programs;

encrypting the data items using one or more keys that are derived from said previous logon authentication of the current computer user;

storing the encrypted data items.

35. A method as recited in claim 34, wherein the storing step comprises storing encrypted data items from different application programs in a common storage area.

36. A method as recited in claim 34, further comprising a step of deriving said one or more keys based on a computer logon password.

38. A method as recited in claim 34, wherein the encrypting step comprises:

encrypting individual data items with item keys;

encrypting the item keys with a master key;

encrypting the master key with a user key that is derived from said previous authentication of the current computer user;

the method further including a step of storing the encrypted item keys and the encrypted master key.

38. A method as recited in claim 34, further comprising the following additional steps:

retrieving encrypted individual data items, encrypted item keys corresponding to the encrypted individual data items, and an encrypted master key;

decrypting the encrypted master key with a user key that is derived from said previous authentication of the current computer user;

decrypting the encrypted item keys with the decrypted master key;

decrypting the encrypted individual data items with the corresponding decrypted item keys.

39. A method as recited in claim 34, comprising a further step of allowing access to individual data items depending on the current computer user.

40. A method as recited in claim 34, comprising a further step of returning requested data items only to authorized requesting application programs.

41. A method as recited in claim 34, comprising the following additional steps:

returning requested data items only to authorized requesting application programs;

authenticating requesting application programs before returning individual data items.

42. A method as recited in claim 34, comprising a further step of presenting user dialogs in response to application program requests for data items, the user dialogs allowing the current computer user to specify passwords to be used by the storage provider to decrypt data items.

43. A method of storing user and application secrets and for protecting them from unauthorized access, comprising the following steps:

receiving individual data items from application programs; encrypting the individual data items with item keys; encrypting the item keys with a master key; encrypting the master key with a user key;

storing the encrypted data items, the encrypted item keys, and the encrypted master key.

44. A method as recited in claim 43, further comprising the following additional steps:

generating item authentication codes for individual data items using item authentication keys;

encrypting the item authentication keys with the master key;

generating key authentication codes for item keys and item authentication keys using the master authentication key;

encrypting the master authentication key with the user key;

storing the item authentication codes, the encrypted item keys, the encrypted item authentication keys, the key authentication codes, and the encrypted master authentication key.

45. A method as recited in claim 43, further comprising the following additional steps:

retrieving encrypted individual data items, the encrypted item keys corresponding to the retrieved encrypted individual data items, and the encrypted master key;

decrypting the encrypted master key with the user key;

decrypting the encrypted item keys with the decrypted master key;

decrypting the encrypted individual data items with the corresponding decrypted item keys.

46. A method as recited in claim 43, further comprising a step of deriving the user key from a user authentication step.

47. A method as recited in claim 43, further comprising a step of deriving the user key from a computer logon code.

48. A method as recited in claim 43, further comprising a step of deriving the user key from a previous authentication of the current computer user.

49. A method as recited in claim 43, further comprising a step of deriving the user key from network authentication of the current computer user.

50. A method of storing user and application secrets and for protecting them from unauthorized access, comprising the following steps:

receiving individual data items from application programs;

encrypting the data items using one or more keys that are derived from one or more user-supplied passwords;

storing the encrypted data items;

retrieving and decrypting the stored encrypted data items in response to requests from application programs.

51. A method as recited in claim 50, comprising a further step of presenting user dialogs in response to application program requests for data items, the user dialogs allowing the current computer user to specify the user-supplied passwords.

52. A method as recited in claim 34, comprising the following additional step:

allowing a user to specify a type of user authentication that will be performed before performing the retrieving and decrypting steps.

53. A method as recited in claim 50, further comprising a step of obtaining a default user-supplied password during a computer logon procedure.

54. A method as recited in claim 50, further comprising a step of obtaining a default user-supplied password during a computer logon procedure, the default user-supplied password comprising a computer logon password.

55. A method as recited in claim 50, comprising the following additional steps:

obtaining a default user-supplied password during a network logon procedure;

obtaining additional user-supplied passwords by presenting user dialogs that allow the current computer user to specify such additional user-supplied passwords.

56. A method as recited in claim 50, comprising a further step of presenting user dialogs in response to application program requests for data items, the user dialogs allowing the current computer user to specify passwords to be used by the storage provider to decrypt data items.

57. A method as recited in claim 50, comprising a further step of allowing access to individual data items depending on the current computer user.

58. A method as recited in claim 50, comprising a further step of returning requested data items only to authorized requesting application programs.

59. A method as recited in claim 50, the encrypting step comprising encrypting the individual data items with item keys, comprising the following additional steps:

receiving individual data items from application programs;

encrypting the item keys with a master key;

encrypting the master key with a user key that is derived from said one or more user-supplied passwords.

60. A method of storing user and application secrets and for protecting them from unauthorized access, comprising the following steps:

receiving individual data items from application programs;

encrypting the data items using one or more keys that are derived from one or more user-supplied passwords;

storing the encrypted data items;

retrieving and decrypting the stored encrypted data items in response to requests from application programs;

allowing access to individual data items depending on the current computer user;

returning requested data items only to authorized requesting application programs.

61. A method as recited in claim 60, wherein an application program performs a step of specifying application programs that are authorized to access respective data items.

62. A method as recited in claim 60, wherein an application program performs a step of specifying computer users that are authorized to access respective data items.

63. A method as recited in claim 60, further comprising a step of authenticating authorized requesting application programs using public key cryptography before returning individual data items.

64. A method as recited in claim 60, wherein the storing step comprises storing encrypted data items from different application programs in a common storage area.

65. A method of storing user and application secrets and for protecting them from unauthorized access, comprising the following steps:

receiving individual data items from different application programs;

encrypting the data items;

storing the encrypted data items from the different application programs in a common storage area.

66. A method as recited in claim 65, comprising a further step of allowing access to individual data items depending on the current computer user.

67. A method as recited in claim 65, comprising a further step of returning requested data items only to authorized requesting application programs.

68. A method as recited in claim 65, comprising the following additional steps:

returning requested data items only to authorized requesting application programs;

authenticating requesting application programs before returning individual data items.

69. A method as recited in claim 65, comprising a further step of presenting user dialogs in response to application program requests for data items, the user dialogs allowing the current computer user to specify passwords to be used by the storage provider to decrypt data items.

70. A method as recited in claim 65, wherein the encrypting step is based on one or more keys that are derived from authentication of the current computer user.

71. A method as recited in claim 65, wherein the encrypting step is based on one or more keys that are derived from a user-supplied code.

72. A method as recited in claim 65, wherein the encrypting step is based on one or more keys that are derived from a computer logon code supplied by the current computer user.

The Patent Office

Application No:	GB 9813901.7	**Examiner:**	David Keston
Claims searched:	1-72	**Date of search:**	30 November 1998

Patents Act 1977
Search Report under Section 17

Databases searched:

UK Patent Office collections, including GB, EP, WO & US patent specifications, in:

UK Cl (Ed.P): G4A (AAP, AMX)

Int Cl (Ed.6): G06F 1/00, 12/14; H04L 9/00, 9/30

Other: Selected publications and Online: COMPUTER, WPI

Documents considered to be relevant:

Category	Identity of document and relevant passage		Relevant to claims
A	EP 0820017 A2	(IBM) - see abstract and fig. 2	7-11,14-16, 23-72
A	EP 0717339 A2	(MICROSOFT) - see abstract and columns 1 - 3	1-72
A	EP 0456386 A2	(INTERNATIONAL COMPUTERS) - see abstract and page 4 lines 26-30	1-72
X	EP 0442839 A2	(IBM) - see abstract and columns 2 and 3	1, 5, 6, 17, 18

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art.
Y	Document indicating lack of inventive step if combined with one or more other documents of same category.	P	Document published on or after the declared priority date but before the filing date of this invention.
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application.